



PATENT  
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of:

CARNEY

Serial No.: 10/629,821

Filed: July 30, 2003

For: NEURAL NETWORK TRAINING

**CLAIM TO PRIORITY**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

The benefit of the filing date of the prior foreign application filed in the following foreign country(ies) is hereby requested and the right of priority provided in 35 U.S.C. §119 is hereby claimed:

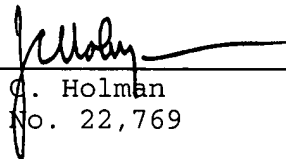
Ireland Application No. 2001/0075 filed 31 January 2001.

In support of this claim, filed herewith is a certified copy of said foreign application.

Respectfully submitted,

JACOBSON HOLMAN PLLC

By: \_\_\_\_\_

  
John C. Holman  
Reg. No. 22,769

400 Seventh Street, N.W.  
Washington, D.C. 20004-2201  
Telephone: (202) 638-6666

Atty. Docket No.: P69049US0  
Date: November 25, 2003  
JCH:crj



Patents Office  
Government Buildings  
Hebron Road  
Kilkenny

I HEREBY CERTIFY that annexed hereto is a true copy of the documents filed in connection with the following patent application:

Application No. 2001/0075

Date of Filing 31/01/2001

Applicant Risk Engines Limited, an Irish company of 1 Orchard Terrace, Grangegorman Road Upper, Phibsboro, Dublin 7, Ireland

Dated this 30 day of July 2003.



An officer authorised by the  
Controller of Patents, Designs and Trademarks.

## REQUEST FOR THE GRANT OF A PATENT

PATENTS ACT, 1992

The Applicant(s) named herein hereby request(s)

X the grant of a patent under Part II of the Act

\_\_\_\_\_ the grant of a short-term patent under Part III of the Act  
on the basis of the information furnished hereunder.

1. Applicant(s)

Name Risk Engines Limited

Address 1 Orchard Terrace  
Grangegorman Road Upper  
Phibsboro  
Dublin 7  
Ireland

Description/Nationality

An Irish company

2. Title of Invention

"Neural network training"

3. Declaration of Priority on basis of previously filed application(s) for same invention (Sections 25 & 26)

Previous filing date

Country in or for  
which filed

Filing No.

4. Identification of Inventor(s)

Name(s) of person(s) believed  
by Applicants(s) to be the inventor(s)

Name: CARNEY, John

Address: 1 Orchard Terrace  
Grangegorman Road Upper  
Phibsboro  
Dublin 7  
Ireland

5. Statement of right to be granted a patent (Section 17(2) (b))

The Applicant derives the rights to the Invention by virtue of a Deed of Assignment dated January 24, 2001

6. Items accompanying this Request – tick as appropriate

- (i)   X   prescribed filing fee (£100.00)
- (ii)   X   specification containing a description and claims  
       specification containing a description only  
  X   Drawings referred to in description or claims
- (iii)        An abstract
- (iv)        Copy of previous application (s) whose priority is claimed
- (v)        Translation of previous application whose priority is claimed
- (vi)   X   Authorisation of Agent (this may be given at 8 below if this Request is signed by the Applicant (s))

7. Divisional Application (s)

The following information is applicable to the present application which is made under Section 24 –

Earlier Application No: .....

Filing Date: .....

8. Agent

The following is authorised to act as agent in all proceedings connected with the obtaining of a patent to which this request relates and in relation to any patent granted -

Name

Address

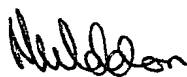
John A. O'Brien & Associates

The address recorded for the time being in the Register of Patent Agents, and currently Third Floor, Duncairn House, 14 Carysfort Avenue, Blackrock, Co. Dublin, Ireland.

9. Address for Service (if different from that at 8)

As above

Signed



JOHN A. O'BRIEN & ASSOCIATES

Date

January 31, 2001



- 1 -

"Neural network training"Introduction

- 5 The invention relates to data processing modelling systems having artificial neural networks.

Applications of such systems include financial prediction and data mining for analysis of customer behaviour.

10

A neural network is a biologically inspired data processing system. A large number of different neural network architectures and training methods exist. By far the most popular neural network for time-series modeling is the back-propagation neural network. The back-propagation neural network is a multi-layered network of processing units and weights. The weights are parameters and together with the processing units (which are usually simple sigmoid transfer functions) they can model a process represented as an input data-set or "training set". The training process used to parameterize the network (i.e. find a good set of weights) is a variant of gradient descent.

20

At present, the generalisation error in such systems arises from bias, variance and noise variance. In applications such as financial prediction the signal-to-noise ratio is often low and thus the (unpredictable) noise variance component is high relative to the (predictable) bias and variance components.

25

It is therefore desirable to reduce bias and variance to achieve good modelling performance.

A number of ensemble techniques have been proposed to reduce variance in neural network regression models. There are different ensemble techniques, and the most

30

popular include some elaboration of “*bagging*” or “*boosting*”. The basic principle of these techniques is to generate multiple versions of a predictor. When predictions from these versions are combined (averaged for example), smoother more stable predictions are generated. When applied to neural networks, these techniques can yield dramatic improvements in prediction performance. This is because neural networks are inherently unstable i.e. small changes in training set and/or parameter selection can produce large changes in performance. Bagging is widely accepted as one of the most popular neural network ensemble techniques. It uses the “bootstrap technique”, a very popular statistical re-sampling technique, to generate multiple training sets and networks for an ensemble. Each ensemble training set is the same size as the original training set, but given that the bootstrap samples data with replacement, individual training samples may appear several times in an ensemble training set while others may be left out. Outputs from the trained networks in a bagged ensemble are combined using a simple average to produce smoother, more stable predictions.

Thus, while ensemble techniques correct for variance, little progress has been made to also correct bias, or to correct for sources of bias that are difficult to detect.

It is therefore an object of the invention to provide a training method and a modelling system to provide correction of both bias and variance and/or to correct for sources of bias. Another object is for such a method and system to estimate average ensemble generalisation error.

#### Statements of Invention

According to the invention, there is provided a method of training a modelling system neural network, the method comprising the steps of:

building stages of bagged ensembles having training sets; and

after each stage adapting the training sets so that bias is identified and compensated for.

- 5 In one embodiment, the method comprises the further steps of estimating generalisation performance at each stage.

In another embodiment, the training sets are generated by sampling with replacement from an original training set.

10

In one embodiment, the step of building stages comprises:-

calculating the number of training vectors in a training set; and

15

setting up initial bootstrap training sets.

In another embodiment, the step of adapting the training set comprises:-

propagating training vectors through a trained ensemble; and

20

storing the ensemble response for each training vector.

In a further embodiment, said step comprises the further steps of:-

25

combining ensembles across all stages trained so far,

calculating the generalisation error; and

30

continuing training while there are significant improvements in the generalisation error.

The invention also provides a modelling system whenever trained in a method as described above.

## 5    Detailed Description of the Invention

The invention will be more clearly understood from the following description of some embodiments thereof, given by way of example only with reference to Fig. 1 which is a flow diagram illustrating operation of a modelling system for financial  
10    prediction.

Referring to Fig. 1 operation of a modelling system of the invention is illustrated. The application in this embodiment is US dollar/Japanese Yen closing price prediction. The inputs are a representative set of historical daily closing prices, and  
15    the output is a closing price prediction. The following are the steps.

1. Retrieve a representative set of historical daily closing prices e.g. 3 years of US Dollar / Japanese Yen daily closing prices.
2. Arrange these into a training set (T) consisting of input vectors and targets.
- 20    3. Set the invention's input parameters i.e. maximum number of epochs (E), number of stages (S), number of networks per stage (B).
4. Trigger the invention by calling the NeuralDVB (E, S, B, T) function.
5. The invention outputs an optimal set of weights (parameters) that can be used to generate a prediction for a target (i.e. US Dollar / Japanese Yen closing price)  
25    when presented with an input vector.

The modelling system's neural network ensemble is trained to have bias and variance close to zero. It thus yields a higher quality output – in this instance a prediction for a US Dollar / Japanese Yen daily closing price that better reflects the true price on



this day. The outputs may be used as decision support indicators for individual traders or as inputs to a systematic trading platform.

5 The modelling system is trained in a method comprising the steps of building stages of bagged ensembles and after each stage the training set is adapted so that bias is identified and compensated for. Training continues until the addition of a new stage does not improve generalization performance. Variance is corrected for at each stage by the bagging.

10 A good estimate of generalization performance is required at each stage. Training sets in a bagged ensemble are generated by sampling with replacement from the original training set. Let us call this training set  $T$ . The probability a training example from  $T$  will not be part of a bootstrap re-sampled training set is approximately  $(1 - 1/N)^N \approx 0.368$ , where  $N$  is the number of training examples in  $T$ . This means that  
15 approximately 37% of the original training examples in  $T$  will not be used for training i.e. they will be out-of-sample. These are used to estimate generalisation error at each stage. The method is described in detail below:

**FUNCTION NeuralDVB** ( $E, S, B, T$ )

20 **RETURNS**  $w$  // Weights for a full set of networks across all stages

**INPUTS**  $E$  // Maximum number of epochs

$S$  // Maximum number of stages

$B$  // Maximum number of networks in an ensemble

$T$  // Original training set

25 // It consists of a set of input vectors and targets i.e.

$$T = \{(t_n, x_n)\}_{n=1}^N$$

// Calculate the number of training vectors in training set

```
 $N \leftarrow |\mathbf{T}|$ 

// Set-up initial bootstrap training sets
FOR  $b=1$  TO  $B$ 
5       $\mathbf{T}_b^* \leftarrow$  Sample with replacement  $N$  times from  $\mathbf{T}$ 
ENDFOR

 $\mathbf{T}^* \leftarrow \{\mathbf{T}_1^*, \dots, \mathbf{T}_B^*\}$ 

10 // Loop to build stages
    $s \leftarrow 1$ 
   REPEAT

       // Build a single stage. Note that variance is corrected for at each stage by the
15       bagging
        $\mathbf{w}_{opt}^{\mathbf{T}_s^*} \leftarrow \text{TrainStage}(N, E, B, \mathbf{T}_s^*)$ 

       // Propagate training vectors through ensemble just trained and
       // store the ensemble responses for each training vector
20        $\mathbf{M} \leftarrow \text{PropStage}(N, s, \mathbf{T}^*, \mathbf{w}_{opt}^{\mathbf{T}_s^*})$ 

       // Combine ensembles across all stages trained so far, calculate the
       // generalisation error
       // and if there is no significant improvement set finished to true
25        $finished \leftarrow \text{CombineStages}(N, s, \mathbf{M}, \mathbf{T}, olderr)$ 

       // If finished is still false then adapt training set and continue
       IF ( $finished = FALSE$ )
            $\mathbf{T}_{s+1}^* \leftarrow \text{AdaptSet}(N, s, \mathbf{M}, \mathbf{T}_s^*)$ 
```

```

// Increment stage
s ← s + 1

5  WHILE ((finished = FALSE) AND (s ≤ S))

    // Set-up return matrix
     $\mathbf{W}^* \leftarrow \{\mathbf{w}_{opt}^{T_1^*}, \dots, \mathbf{w}_{opt}^{T_S^*}\}$ 

10  RETURN ( $\mathbf{W}^*$ )

FUNCTION TrainStage ( $N, E, S, B, \mathbf{T}^*$ )
RETURNS  $\mathbf{w}_{opt}^{T^*}$  // Optimal set of weights for this stage
15 INPUTS  $N, E, S, B$  // As above
            $\mathbf{T}^*$  // Set of bootstrap training set for this stage

    // Copy training sets for this stage into individual sets
     $\{\mathbf{T}_1^*, \dots, \mathbf{T}_S^*\} \leftarrow \mathbf{T}^*$ 

20    // Compute ensemble generalisation error estimates for each training
    //example. Note that  $\gamma_n^h$  is a variable that indicates whether training
    //example  $n$  is out-of-sample for bootstrap training set  $\mathbf{T}_h^*$  or not;  $\gamma_n^h = 1$  if it
    //is and  $\gamma_n^h = 0$  if it is not. Also, note that  $\phi(\mathbf{x}_n; \mathbf{w}_e^{T_i})$  is used to represent the
25    //response of a neural network, given input vector  $\mathbf{x}_n$  and weights trained
    //for  $e$  epochs using training set  $\mathbf{T}_i^*$ 

    FOR  $n=1$  TO  $N$ 
        FOR  $e=1$  TO  $E$ 

```

Compute:

$$G_e^n \leftarrow \left( t_n - \frac{\sum_{h=1}^B \gamma_n^h (\phi(\mathbf{x}_n; \mathbf{w}_e^{T_h}))}{\sum_{h=1}^B \gamma_n^h} \right)^2$$

ENDFOR

ENDFOR

5

// Aggregate the ensemble generalisation error estimates for each training //  
example to produce an estimate for the average ensemble generalisation //  
error

FOR  $e=1$  TO  $E$

10

Compute:

$$A_e \leftarrow \frac{1}{N} \sum_{n=1}^N G_e^n$$

ENDFOR

// Find the best value for  $e$  for each network in the ensemble

15

Compute:

// Find the optimal value for  $e$  i.e. the value for  $e$  that  
//minimises the average ensemble generalisation error. The  
//corresponding networks are chosen as the optimal set for  
//the ensemble.

20

$$e_{opt} \leftarrow \arg \min_e (A_e)$$

$$\mathbf{W}_{opt}^{T*} \leftarrow \mathbf{w}_{\mathcal{E}_{opt}}^{T*}$$

RETURN ( $\mathbf{w}_{opt}^{T*}$ )

25 FUNCTION PropStage ( $N, s, \mathbf{T}_s^*, \mathbf{W}_{opt}^{T*}$ )

RETURNS  $\mathbf{M}$

**INPUT**  $N, s, T^*, w_{opt}^{T^*}$  // As above

// Compute ensemble (bagged) outputs for each training example for this  
//stage. These will be used later to adapt the training set

5 **FOR**  $n=1$  **TO**  $N$

    Compute:

$$M_n^* \leftarrow \frac{\sum_{h=1}^B \gamma_n^h (\phi(x_n; w_{opt}^{T^*}))}{\sum_{h=1}^B \gamma_n^h}$$

**ENDFOR**

10 **RETURN** ( $M$ )

**FUNCTION** **CombineStages** ( $N, s, M, T, olderr$ )

15 **RETURNS** *finished*

**INPUT**  $N, s, M, T, olderr$  // As above

// Set new variable as upper bound on number of stages so far  
 $numstages \leftarrow s$

20

// Sum ensemble outputs across stages

**FOR**  $n=1$  **TO**  $N$

$$S_n \leftarrow \sum_{j=1}^{numstages} M_n^j$$

**ENDFOR**

25

// Calculate staged ensemble generalisation error

$$newerror \leftarrow \frac{1}{N} \sum_{n=1}^N (t_n - S_n)^2$$

```
// If no improvement finish training
IF  $s = 1$ 
     $olderr \leftarrow newerror$ 
5    ELSE IF ( $newerror > (\delta * olderr)$ )
        //  $\delta$  is a tuning variable that sets the strictness of the stopping condition
        // for adding new stages
         $finished \leftarrow 1$ 
    ELSE IF ( $newerror < olderr$ )
10         $olderr \leftarrow newerror$ 

RETURN ( $finished$ )

15 FUNCTION AdaptSet ( $N, s, M, T^*$ )
RETURNS  $T^*$ 
INPUT  $N, s, M, T^*$  // As above

20 // Set new variable as upper bound on number of stages so far
 $numstages \leftarrow s$ 

// Sum ensemble outputs across stages
FOR  $n=1$  TO  $N$ 
25      $S_n \leftarrow \sum_{j=1}^{numstages} M_n^j$ 
ENDFOR

// Adapt training set
FOR  $n=1$  TO  $N$ 
```

$$t_{n,s+1}^* \leftarrow t_{n,s}^* - S_n$$

ENDFOR

RETURN ( $T_{s+1}^*$ )

5

Note that NeuralDVB returns a set of weights ( $w'$ ) for a neural network ensemble that has a bias and variance close to zero. These weights (which are simply floating point numbers) can be used to generate a prediction for any input vector drawn from the same probability distribution as the training set input vectors.

10

It will be appreciated that the invention provides the following improvements over the art:

1. It explicitly corrects for bias and variance in neural networks.
- 15 2. It corrects for sources of bias that are difficult to detect and are not reflected in the average mean-squared generalisation error. For example, some time-series data such as financial data can have a dominant directional bias. This is problematic as it can cause neural network models to be built that perform well based on the average mean-squared error but poorly when predicting a directional change that is not well represented in the training data. The invention automatically corrects for this bias (along with usual sources of bias) despite it not being reflected in the average mean-squared generalisation error
- 20 3. It uses an early-stopping based method to estimate average ensemble generalisation error. Good estimates of generalisation performance are critical to success.
- 25

The invention is not limited to the embodiments described but may be varied in construction and detail.

Claims

1. A method of training a modelling system neural network, the method comprising the steps of:  
5  
building stages of bagged ensembles having training sets; and  
after each stage adapting the training sets so that bias is identified and compensated for.  
10
2. A method as claimed in claim 1, wherein the method comprises the further steps of estimating generalisation performance at each stage.
3. A method as claimed in claims 1 or 2, wherein the training sets are generated  
15 by sampling with replacement from an original training set.
4. A method as claimed in any preceding claim, wherein the step of building stages comprises:-  
20 calculating the number of training vectors in a training set; and  
setting up initial bootstrap training sets.
5. A method as claimed in any preceding claim, wherein the step of adapting the  
25 training set comprises:-  
propagating training vectors through a trained ensemble; and  
storing the ensemble response for each training vector.  
30



6. A method as claimed in claim 5, wherein said step comprises the further steps of:-

combining ensembles across all stages trained so far,

- 5 calculating the generalisation error; and

continuing training while there are significant improvements in the generalisation error.

- 10 7. A modelling system neural network training method substantially as hereinbefore described.

8. A modelling system comprising a neural network whenever trained in a method as claimed in any preceding claim.

15

9. A computer program product comprising software code for performing the steps of any of claims 1 to 7 when executing on a digital computer.

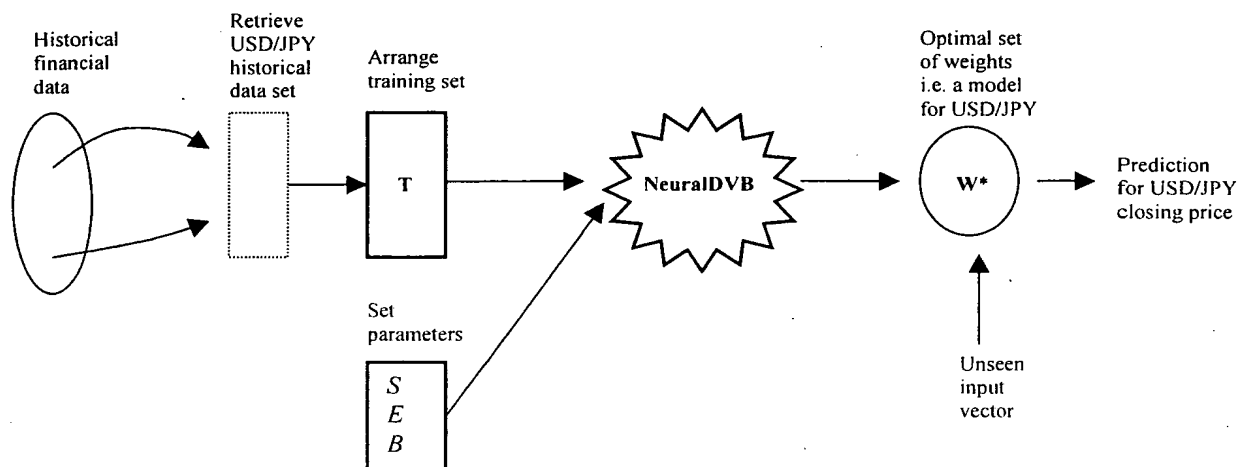


Fig. 1